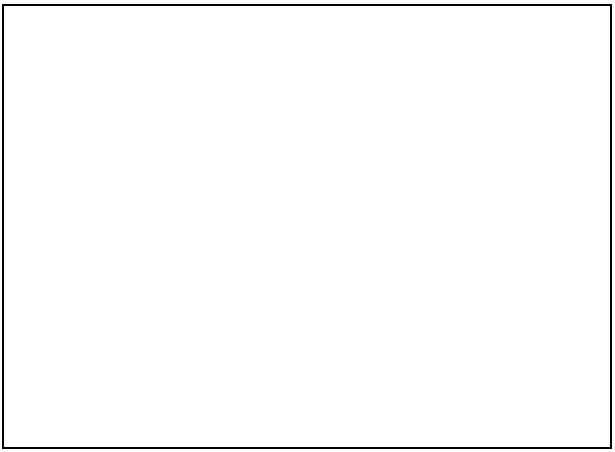
SIO-DLL

Serial I/O DLL



User Manual

Serial DLL Software

User Manual

Document Part N° 127-157
Document Reference DLL\SER_DLL\..\127_157.doc
Document Issue Level 1.2

Manual covers software version 1

All rights reserved. No part of this publication may be reproduced, stored in any retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopied, recorded or otherwise, without the prior permission, in writing, from the publisher. For permission in the UK contact Blue Chip Technology.

Information offered in this manual is correct at the time of printing. Blue Chip Technology accepts no responsibility for any inaccuracies. This information is subject to change without notice.

All trademarks and registered names acknowledged.

Blue Chip Technology Ltd.
Chowley Oak, Tattenhall,
Chester,
Cheshire, CH3 9EX.
Telephone : (01829) 772000 Facsimile : (01829) 772001.

Amendment History

Issue Level	Issue Date	Author	Amendment Details
1.0	11/9/94	-	Document Reference was BCTSIO10.doc
1.1	22/1/96	EGW	Revised front sheet. Reformatted to A5
1.2	5/6/96	PDJ	New address added
1.3	02/7/97	SEJ	New front cover

INTRODUCTION **1**

FEATURES OF THE SERIAL DLL FOR WINDOWS 1
TYPICAL APPLICATIONS 2
THE FUNCTIONS 2
THE PACKAGE 2
SIOTEST 3
REQUIREMENTS 3

INSTALLATION **4**

SOFTWARE INSTALLATION 4
HARDWARE INSTALLATION 4
Hardware Interrupts *4*
Sharing IRQ 3 and IRQ4 ..... *5*

USING THE SERIAL DLL **7**

AN EXAMPLE USING C 7
AN EXAMPLE USING VISUAL BASIC 8

FUNCTION REFERENCE **9**

FUNCTION QUICK REFERENCE 9
DETAILED FUNCTION REFERENCE 10
SerialCheckPortNo *10*
SerialGetBuffer *10*
SerialGetByte *11*
SerialInstallPort *11*
SerialPortCount *14*
SerialPutBuffer *14*
SerialPutByte *15*
SerialReadModemStatus *16*
SerialReadInterruptStatus *17*
SerialRemoveAllPorts *18*
SerialRemovePort *18*
SerialRxClear *19*
SerialRxCount *19*
SerialRxFree *20*
SerialSendBreak *21*
SerialSetDTR *21*
SerialSetInterruptAddress *22*
SerialSetRTS *23*
SerialSetOut1 ..... *24*

SerialStart *24*
SerialStop *25*
SerialTxReady *26*
SerialTxEmpty *26*
SerialVersion *27*
SerialWriteModemControl *27*
SerialReadModemControl *29*
RETURN VALUES AND ERROR CODES 30

SIOTEST 31

RUNNING SIOTEST 31
SIOTEST FUNCTIONS 34

SYSTEM REQUIREMENTS 44

PERFORMANCE CONSIDERATIONS 44

SUMMARY OF THE DISTRIBUTION DISK 47

NOTES ABOUT SPECIFIC CARDS 49

INTRODUCTION

The Blue Chip Technology Serial DLL (BCTSIO) for Windows is designed for programmers who wish to add interrupt driven serial communications to their Windows programs for IBM/PC/XT/ATs and 100% compatibles. ***BCTSIO*** is particularly suited for systems requiring more serial ports than normally supported by Windows.

In addition to ***BCTSIO*** a fully comprehensive test program is supplied complete with source code.

BCTSIO is Windows DLL (Dynamic Link Library) containing a suite of interrupt driven RS232 device drivers. By making calls to ***BCTSIO*** the users program can configure up to 64 serial ports and transfer data through any of them.

BCTSIO receives any characters from the serial port into character input queue (the size can be set by the user from 16 to 32000 bytes), 1 queue per port. This means that the user program may ignore any characters received at the serial port for a while without losing any characters.

Features Of The Serial DLL For Windows

- Simple to use Windows based interrupt driven RS232 communications
- Programming language independent (Visual Basic, C, Pascal, Assembler etc.)
- Supports up to 64 serial ports
- Support serial ports at any I/O address (The address may be user defined)
- All receive channels are fully buffered (up to 30Kbytes per port)
- Support any of the standard baud rates, bits per character and number of stop bits
- User specified hardware interrupt channels (Any port may be assigned to any interrupt)
- RS422/RS485 Support for cards that enable the Transmitter using one of the UART control lines
- Graphical User interface allowing all the functions, including the transmission and monitoring of serial data, to be exercised without the need for any programming (Complete Source Code supplied)

Typical Applications

- Use with the Blue Chip Technology Range of serial cards including the MSC8 eight port card
- Advanced programming applications
- Systems Integration using serial data links
- Introduction to Serial programming under Windows
- Data Acquisition

The Functions

The following functions are provided by ***BCTSIO***:

- Install and configure a communications port
- Remove an installed port
- Remove all installed ports
- Get a character from the specified port
- Get a buffer from the specified port
- Send a byte to the specified port
- Start the port hardware interrupts
- Stop the port hardware interrupts
- Send a buffer to a specified port
- Return the number of characters in the receive buffer
- Return the free space in the receive buffer
- Check to see if the port transmitter is ready for another character
- Clear the receive buffer of any characters
- Read the state of the port's modem lines
- Set the state of the ports modem lines
- Transmit a break to the specified port
- Return the current Serial DLL version number
- Check to see if the current port id is valid
- Return the total number of characters received
- Set/Reset DTR (RS422/RS485 transmitter enable)
- Set/Reset RTS (RS422/RS485 transmitter enable)
- Set/Reset OUT1 (RS422/RS485 transmitter enable)
- Check the Transmitter status

The Package

The ***BCTSIO*** package consists of the following items:

BCTSIO.DLL	The Serial I/O Dynamic Link Library
SIOTEST.EXE	Serial I/O DLL Interactive User Interface including complete source code.
SETUP.EXE	Fully interactive Set-up Utility
BCTSIO.H	Header file for C programs using the Serial DLL.

SIOTEST

SIOTEST has a Graphical User Interface which allows the user to exercise all the functions of the Serial DLL and monitor the data transfer on all of the serial ports without the need for any programming.

The complete source code for SIOTEST is supplied (in Visual Basic) which provides a complete set of examples showing the usage of each ***BCTSIO*** function. This may also provide a starting point for your own project.

Requirements

BCTSIO requires that the hardware is compatible with the following Blue Chip Technology communication adapters:

- MSC8 2, 4, 6 and 8 channel versions
- SIO2
- SPIO (SIO2P)
- SIO4d
- Any IBM PC compatible Asynchronous Communications Adapter using the INS8250 UART or equivalent and that pin OUT2 is used to enable the interrupt output.

BCTSIO is designed to operate with Windows version 3.1 in enhanced mode. To share an interrupt between multiple ports the hardware must have been specifically design to do so, e.g. the Blue Chip Technology MSC8 card.

INSTALLATION

Software Installation

To install ***BCTSIO*** follow the procedure below:

- Run Windows
- Place the ***BCTSIO*** installation floppy diskette in drive A or drive B
- Run SETUP.EXE on the installation disk by selecting File/Run from the Program Manager menu
- Select the directory to install ***BCTSIO*** in, the default is C:\BCTSIO
- This directory will contain the ***BCTSIO*** and the SIOTEST source files
- A Windows group SIOTEST will be created containing an ICON for SIOTEST
- Double clicking on the SIOTEST icon will run the ***BCTSIO*** test program SIOTEST.

Hardware Installation

There are a few important points to consider when installing multiple port serial cards under Windows.

Hardware Interrupts

You should only use hardware interrupts IRQ0-IRQ15 if they are not already being used by your computer hardware or Windows.

These are typically used as follows (your system may be different):

IRQ0	Timer
IRQ1	Keyboard
IRQ2	2nd interrupt controller
IRQ3	Serial channel 2 (COM2)
IRQ4	Serial channel 1 (COM1)
IRQ5	Second Parallel Printer (often not used)
IRQ6	Floppy Disk
IRQ7	First Parallel Printer (often not used)
IRQ8	Real time clock
IRQ9	Redirected to IRQ2
IRQ10	Reserved
IRQ11	Reserved
IRQ12	Reserved
IRQ13	Maths Unit
IRQ14	Fixed disk controller
IRQ15	Reserved

Sharing IRQ 3 and IRQ4

Windows 3.1 provides a communications driver to support serial ports COM1 on IRQ4 and COM2 on IRQ3. This driver only allows COM1 to use IRQ4 and COM2 to use IRQ3, any attempt to support more than one serial port on either IRQ will be blocked by Windows.

the Serial DLL will support COM1 on IRQ3 and COM2 on IRQ4 with no additional considerations. However if more than one port is to be installed on either IRQ3 or IRQ4 the Windows communications drivers must be disabled. To disable the windows serial communications driver make the following modifications to the SYSTEM.INI file in your windows directory:

where you see the entry
comm.drv=comm.drv
place a semicolon before it, i.e.:
;comm.drv=comm.drv

This will remove the current Windows communications drivers.

After removing the Windows communications drivers there would appear to be no limit on how many serial ports may be configured to use it.

In addition Windows also appears to monitor ports at COM3 (03E8) and COM4 (02E8) and therefore unless the comm driver is disabled as described above, these addresses should be avoided.

Subject to already being occupied by existing hardware, the following addresses may be tried without disabling the Windows comm drivers:

280, 288, 290, 298, 2A0, 2A8, 2B0, 2B8, 1A0, 1A8, 1B0 and 1B8.

USING THE SERIAL DLL

BCTSIO is a Windows DLL (Dynamic Link Library) providing 21 functions to support serial communications, especially for the support of multiple serial port cards. ***BCTSIO*** functions may be called by any Windows based application that can make DLL calls including 'C',Visual Basic, Pascal, Excel etc.

An Example Using C

```
#include "bctsio.h"                    /* Include the BCTSIO definitions */

...
int PortNo;                            /* The current port number */
int result;                            /* The result */

SerialStop();                          /* Ensure interrupts are stopped */
PortNo=SerialInstallPort(0x3F8,1000,4,9600L,0,8,1 );   /* Install Port */
if ( PortNo < 0 )
        ... report error here and exit ....      /* If error report it */
result =SerialStart();                 /* Start Interrupts */
if(result < 0 )
        ... report error here and exit ....      /* If error report it */
result=SerialPutBuffer(PortNo,"1234567890",10);
                                       /* Transmit the message '1234567890' */
 if(result < 0 )
        ... report error here and exit ....      /* If error report it */
while((result = SerialGetByte(PortNo))!=0x1b)
                                       /* Read and display  received characters */
        print_character(result);               /* until Escape character detected */
SerialStop();                          /* Stop interrupts */
SerialRemovePort(PortNo);              /* Remove the port from the operations */
```

An Example Using Visual Basic

The Project (MAKE) file should include the file BCTSIO.TXT which has a declaration for each of the ***BCTSIO*** functions.

```
Dim PortNo as Integer                   ' The current port number
Dim Result as Integer                   ' The result

result = SerialStop()                   ' Ensure interrupts are stopped
PortNo=SerialInstallPort(&H3F8,1000,4,9600,0,8,1 )    ' Install Port
if  PortNo < 0 then
        ... report error here and exit ....     ' If error report it
result =SerialStart()                   ' Start Interrupts
if result < 0 then
        ... report error here and exit ....     ' If error report it
result=SerialPutBuffer(PortNo,"1234567890",10)
                                        ' Transmit the message '1234567890'
 if result < 0 then
        ... report error here and exit ....     ' If error report it
Do
        result = SerialGetByte(PortNo)  ' Read and display any received
                                        characters
        if result = chr$(27) then Exit Do
        print_character(result);        ' until an Escape character is detected
Loop
result= SerialStop()                    ' Stop interrupts
result=SerialRemovePort(PortNo)         ' Remove the port from the operations
```

FUNCTION REFERENCE

This section contains an alphabetic reference to the functions contained in ***BCTSIO***.

Function Quick Reference

FUNCTION	SHORT DESCRIPTION
SerialCheckPortNo	Check port number is valid
SerialGetBuffer	Copy received characters into a buffer
SerialGetByte	Get a received character
SerialInstallPort	Install a port
SerialPortCount	Count the number of installed ports
SerialPutBuffer	Transmit a buffer of characters
SerialPutByte	Transmit a character
SerialReadInterruptStatus	Read the interrupt status register
SerialReadModemControl	Read the modem control register
SerialReadModemStatus	Read the modem status lines
SerialRemoveAllPorts	Remove all installed ports
SerialRemovePort	Remove an installed port
SerialRxClear	Delete all the received characters
SerialRxCount	Count the number of received characters
SerialRxFree	Count the free space in the receiver
SerialRxIntCount	Count the number of all characters received
SerialSendBreak	Transmit a break
SerialSetDTR	Set or Reset the DTR line
SerialSetInterruptAddress	Set the Address of the Interrupt Address Register
SerialSetRTS	Set or Reset the RTS line
SerialSetOut1	Set or Reset the OUT1 line
SerialStart	Start the Serial running
SerialStop	Stop the Serial running
SerialTxReady	See if the transmitter is ready for another byte
SerialTxEmpty	See if the transmitter is completely empty
SerialVersion	Return the current version of the Serial DLL
SerialWriteModemControl	Set the modem control lines

Detailed Function Reference

This section describes each of the BCTSIO functions in turn.

SerialCheckPortNo

Definition

C int SerialCheckPortNo (int)

Visual Basic SerialCheckPortNo(ByVal PortNo as Integer) as Integer

Parameters

PortNo The number returned from calling SerialInstallPort to install a port

Return Value 1 If PortNo is valid, -1 otherwise

Description

SerialCheckPortNo is called to check to see if a PortNo is valid and has been installed using SerialInstallPort

Example

```
result = SerialCheckPortNo(3)
```

In the example result = 1 if PortNo 3 is valid, i.e. a previous call to SerialInstallPort returned 3. Otherwise result = -1.

SerialGetBuffer

Definition

C int SerialGetBuffer(int PortNo, LPSTR *Buffer, unsigned int Buflen)

Visual Basic SerialGetBuffer(ByVal PortNo as Integer, Buffer as String,

Parameters

PortNo The Port number as returned from SerialInstallPort

Buffer The address of the buffer to place the data in

Buflen The maximum number of characters to copy to the buffer 0 means copy all of the available characters from the buffer, ensure that there is enough space to receive them.

Return Value The number of characters transferred or -2 if the PortNo was invalid

Description

Copy up to the specified number of characters from the receiver buffer to the specified buffer. If less then the specified number of characters were in the receive buffer then only the number of characters available are copied.

Example

```
len =SerialGetBuffer(4, buffer,100)
```

Copy up to 100 characters from the receive buffer on port 4.

SerialGetByte

Definition

C	int SerialGetByte(int PortNo);
Visual Basic	SerialGetByte(PortNo as Integer) as Integer

Parameters

PortNo	The port number returned from SerialInstallPort when installing the port
Result	The code of the next character in the receive buffer or -1 if no characters are available

Description

Call SerialGetByte in order to get the next character from the receive buffer.

Example

```
byte = SerialGetByte(0)
```

Get the next byte from port 0, if byte = -1 then none were available.

SerialInstallPort

Definition

C	int SerialInstallPort(unsigned int PortAddress, int Buflen, int IrqNo, long Baud, int Parity, int bitsPerChar, int StopBits);
Visual Basic	SerialInstallPort(PortAddress as Integer, Buflen as Integer, IrqNo as Integer, Baud as Long, Parity, as Integer, BitsPerChar as Integer, StopBits as Integer) as Integer

Parameters

PortAddress Serial Port Address if greater than 15 or 1-16 Specifies a particular 'COM' port.

No.	Port	Address
1	COM1	3f8H †
2	COM2	2f8H †
3	COM3	3E8H †
4	COM4	2e8H †
5	COM5	280H
6	COM6	288H
7	COM7	290H
8	COM8	298H
9	COM9	2a0H
10	COM10	2a8H
11	COM11	2b0H
12	COM12	2b8H
13	COM13	1a0H
14	COM14	1a8H
15	COM15	1b0H
16	COM16	1b8H

† See section on Sharing IRQ 3 and IRQ4

Buflen Length of buffer that receives input characters. The longer the buffer the less often the program needs to check the number of characters in the buffer. Minimum 16 bytes. Maximum 32000 bytes.

IrqNo The Hardware interrupt request line that the serial port is connected to for receive character interrupts. Minimum 0. Maximum 15

Baud The baud rate that the characters are transmitted and received at. Permitted values are: 50, 75, 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200

Parity The parity settings for the transmitted and received characters. Permitted values are:

No Parity 0 or 'N'

Odd Parity	1 or 'O'
Even Parity	2 or 'E'

BitsPerChar The number of data bits per transmit and receive character. Permitted values are: 5,6,7 or 8

StopBits The number of stop bits per transmit and receive character. Permitted values are: 1 or 2.

PortNo		
	>0	Channel No, port installed successfully
	-3	Too many port installed, max =64.
	-4	The port at the specified address has already been installed
	-5	Not enough memory to allocate receive buffer
	-6	Interrupts running, stop with SerialStop first
	-10	Invalid baud rate specified
	-11	Invalid parity specified
	-12	Invalid number of stop bits specified
	-13	Invalid number of bits/character specified

Description

Before a port can be used for communications it must be initialised and allocated to a hardware interrupt (IRQ) line. SerialInstallPort is used to initialise the port, allocate the port to a particular IRQ and allocate a receive buffer. The return value from the called returns either an error code or a port number (or handle) which identifies the particular port. This port number is used with many of the other functions to identify the port.

A port may be removed from the communications system by a call to the function SerialRemovePort.

The communications routines must be inactive when SerialInstallPort is called, so it either must not have been started with SerialStart or SerialStop must be called first.

Example

```
result =SerialStop()
portno =SerialInstallPort(0x3f8,1000,4,9600,2,7,1)
result =SerialStart()
```

In the example:

The interrupt and communications system is stopped
The serial port at address 3F8 hex (COM1) IRQ 4 is set to 9600 baud with Even Parity, 7 bits per character and 1 stop bit. The receive buffer length is set to 1000 bytes long.
The interrupt system and communications system is started up.

SerialPortCount

Definition

C int SerialPortCount(void)
Visual Basic SerialPortCount() As Integer

Parameters

Return Value The number of ports that are currently installed in the DLL.

Description

Returns the number of ports that are currently installed in the Serial DLL.

Example

```
result=SerialPortCount()
```

Returns the number of ports that are currently installed in the Serial DLL.

SerialPutBuffer

Definition

C int SerialPutBuffer(int PortNo, LPSTR *Buffer, unsigned int Buflen);
Visual Basic SerialPutBuffer(ByVal PortNo as Integer, Buffer as string, ByVal Buflen as Integer) as Integer

Parameters

PortNo The port number returned from SerialInstallPort when installing the port
Buffer The buffer containing the characters to send
Buflen The number of characters to transmit from the buffer
Return Value 0 The buffer was successfully transmitted
-2 Invalid PortNo

Description

Buflen characters from Buffer are transmitted to port PortNo.
Note: If the port transmitter is not ready SerialPutBuffer will wait until it is before sending the next character

Example

```
result = SerialPutBuffer(1,"Hello",5)
```

Transmit the message "Hello" to port 1.

SerialPutByte

Definition

C int SerialPutByte(int PortNo, int Byte)
Visual Basic SerialPutByte(PortNo as Integer, Byte as Integer) as Integer

Parameters

PortNo	The Port number as returned from the called to SerialInstallPort	
Byte	The 8 bit value to send to the specified port	
Result	0	If successfully transmitted
	-2	If PortNo invalid

Description

The specified byte is transmitted out of the specified port if that port is not already busy transmitting another byte. If it is busy it will wait until the previous byte has been transmitted before sending the byte.

Example

```
result = SerialPutByte(1, 'A');
```

In the example the letter 'A' is transmitted out of port number 1.

SerialReadModemStatus

Definition

C	int SerialReadModemStatus(int PortNo)
Visual Basic	SerialReadModem(PortNo as Integer) as Integer

Parameters

PortNo	The Port number as returned from the called to SerialInstallPort
Return value	-2 If PortNo is invalid
	Status Modem Status, see description below

Description

This function is used to read the state of the modem control lines (inputs) which are reflected in the modem status register. The bits are encoded as follows:

Bit	
0	Delta CTS
1	Delta DSR
2	Trailing edge of RI
3	Delta DCD
4	Complement of CTS input
5	Complement of DSR input
6	Complement of RI input
7	Complement of DCD input

Bits 8 to 15 are not used.

For further details refer to the data sheets on the serial card you are using and the manufacturers data sheets for the 8250 (or equivalent) UART I.C.

Example

```
status=SerialReadModemStatus(1)
```

Read the status of the modem control lines on port 1.

SerialReadInterruptStatus

Definition

C int SerialReadInterruptStatus(int IrqNo)
Visual Basic SerialReadInterruptStatus(IrqNo integer) as Integer

Parameters

IrqNo The IRQ number of the Interrupt Status Register as set in the SetInterruptAddress function. (MSC4/6/8 card only)

Return value -2 If PortNo is invalid
Status Interrupt Status, see description below

Description

This function is used to read the interrupt status from the MSC4/6/8 card. The bits are encoded as follows:

Bit	
0	Channel 1
1	Channel 2
2	Channel 3
3	Channel 4
4	Channel 5
5	Channel 6
6	Channel 7
7	Channel 8

Bits 8 to 15 are not used.

A bit set indicates that a channel has a pending interrupt.
A bit cleared indicates no interrupt pending on that channel.
For further details refer to the MSC 4/6/8 data sheets.

Example

```
status=SerialReadInterruptStatus(5)
```

Read the status of the Interrupt Status Register monitoring interrupts on IRQ 5.

SerialRemoveAllPorts

Definition

C int SerialRemoveAllPorts(void)
Visual Basic SerialRemoveAllPorts() as Integer

Parameters

Return value 0 All ports remove from interrupts successfully
-6 Interrupts running, call SerialStop first

Description

Remove all the previously installed ports from operating under the Serial DLL.

Example

```
result=SerialRemoveAllPorts()
```

SerialRemovePort

Definition

C int SerialRemovePort(int PortNo)
Visual Basic SerialRemovePort(ByVal PortNo) as Integer

Parameters

PortNo The port number returned from SerialInstallPort when installing the port
Return value 0 All ports remove from interrupts successfully
-2 PortNo is invalid
-6 Interrupts running, call SerialStop first

Description

Remove the previously installed port Portno from operating under ***BCTSIO***.

Example

```
result = SerialRemovePort(6)
```

Remove the previously installed port 6 from ***BCTSIO*** operations.

SerialRxClear

Definition

C int SerialRxClear(int PortNo);
Visual Basic SerialRxClear(ByVal PortNo as Integer) as Integer

Parameters

PortNo The port number returned from SerialInstallPort when installing the port
Return value 0 Receive buffer cleared successfully
-2 PortNo is invalid

Description

Delete all of the characters from the receive buffer for the specified port.

Example

```
result = SerialRxClear(13)
```

Remove all the characters from the receive buffer for port 13.

SerialRxCount

Definition

C int SerialRxCount(int PortNo);
Visual Basic SerialRxCount(ByVal PortNo as Integer) as Integer

Parameters

PortNo The port number returned from SerialInstallPort when installing the port
Return value n>=0 Number of characters in receive buffer
-2 PortNo is invalid

Description

Return the number of characters ready to be read from the receive buffer for the specified port.

Example

```
result = SerialRxCount(2);
```

Return the number of characters in the receive buffer for port 2.

SerialRxFree

Definition

C int SerialRxFree(int PortNo)
Visual Basic SerialRxFree(ByVal PortNo as Integer) as Integer

Parameters

PortNo The port number returned from SerialInstallPort when installing the port
Return value n>=0 Amount of free space remaining in receive buffer
-2 PortNo is invalid

Description

Return the amount of free space remaining in the receive buffer. This is equal to the maximum length of the receive buffer minus the number of characters in the receive buffer.

Example

```
result = SerialRxFree(8);
```

Return the number of free bytes left in the receive buffer for port 8.

SerialRxIntCount

Definition

C long SerialRxIntCount(void);
Visual Basic SerialRxIntCount() as Long

Parameters

Return value The number of characters received in total since the last call to SerialStart.

Description

This function returns the number of receiver interrupts, which is equal to the number of characters received for all ports in total, since the last call to SerialStart.

Example

```
result = SerialRxIntCount();
```

Return the number of received characters since the last call to SerialStart

SerialSendBreak

Definition

C int SerialSendBreak(int PortNo);

Visual Basic SerialSendBreak(ByVal PortNo As Integer) As Integer

Parameters

PortNo The port number returned from SerialInstallPort when installing the port

Return value 0 Break sent successfully
-2 PortNo is invalid

Description

This function transmits a break to the specified port. Normally the transmit data line is sending a 'marking' signal indicating no data is present, A break is a 'spacing' signal which lasts for longer than a valid character pattern.

Example

```
result = SerialSendBreak(4);
```

Transmits a break on port 4.

SerialSetDTR

Definition

C int SerialSetDTR(int PortNo,int State);

Visual Basic SerialSetDTR(ByVal PortNo As Integer ,ByVal State As Integer) As Integer

Parameters

PortNo — The port number returned from SerialInstallPort when installing the port

State — 0 = Set RS232 DTR line to -12v
1 = Set RS232 DTR line to +12v

Description

This function allows the DTR line to be either set or reset. This line may also be used to enable or disable the transmitter enable on cards supporting this feature.

Example

```
SerialSetDTR(3,1)
```

Set the RS232 DTR line to +12v

SerialSetInterruptAddress

Definition

C int SerialSetInterruptAddress(int IrqNo, int Address)

Visual Basic SerialSetInterruptAddress(ByVal IrqNo As Integer , ByVal Address As Integer) As Integer

Parameters

Address — The address that the Interrupt Status Register is set to on the MCS-4/6/8 card when configured for 'Common Interrupt' Mode.

IrqNo — The hardware interrupt request line that the Interrupt Status Register is assigned to. Minimum 0. Maximum 15

Description

This function will only operate in conjunction with the MSC-4/6/8 range of cards. Normally it is not possible for more than one serial port to share the same interrupt line. However the MSC-4/6/8 range of cards allow up to 8 ports to share one interrupt line.

In order to do this an Interrupt Status Register is used in order to identify which port generated the interrupt. This function is used to specify the address of the Interrupt Status Register and the corresponding hardware

interrupt number. All ports specified as using the same IrqNo as the Interrupt Status Register will be assumed to be sharing interrupts using the 'Common Interrupt' Mode.

The order that the serial ports have been declared is important. The first port to have been allocated to the specified IRQ will be assumed to be channel 1 on the MSC card, the second port to channel 2 etc.

Example

```
SerialInterruptAddress(0x1F0,5)
portno0 =SerialInstallPort(0x3f8,1000,5,9600,2,7,1)
portno1 =SerialInstallPort(0x2f8,1000,5,9600,2,7,1)
portno2 =SerialInstallPort(0x3e8,1000,5,9600,2,7,1)
portno3 =SerialInstallPort(0x2e8,1000,5,9600,2,7,1)
```

Set the address of the Interrupt Status Register on the MSC-4/6/8 card to 0x1F0 serving interrupts using IRQ 5. portno0 will be assigned to channel 1, portno1 will be assigned to channel 2 etc.

SerialSetRTS

Definition

C int SerialSetRTS(int PortNo,int State)

Visual Basic SerialSetRTS(ByVal PortNo As Integer ,ByVal State As Integer) As Integer

Parameters

PortNo	The port number returned from SerialInstallPort when installing the port
State	0 = Set RS232 RTS line to -12v 1 = Set RS232 RTS line to +12v

Description

This function allows the RTS line to be either set or reset. This line may also be used to enable or disable the transmitter enable on cards supporting this feature.

Example

```
SerialSetRTS(8,0)
```

Set the RS232 RTS line to -12v

SerialSetOut1

Definition

C int SerialSetOut1(int PortNo,int State)

Visual Basic SerialSetOut1(ByVal PortNo As Integer, ByVal State As Integer) As Integer

Parameters

PortNo The port number returned from SerialInstallPort when installing the port

State 0 = Set OUT1 to a Logic 1
1 = Set OUT1 to a Logic 0

Description

This function allows the OUT1 line to be either set or reset. This line may also be used to enable or disable the transmitter enable on cards supporting this feature.

Example

```
SerialSetOut1(15,0)
```

Disable the transmitter on RS422/485 port 15.

SerialStart

Definition

C int SerialStart(void);

Visual Basic SerialStart() as Integer

Parameters

Return value 0 = Successfully started interrupts
-8 = Interrupts already started
-16 = Interrupt Status Register Address not set for an IRQ where more than one port has been allocated

Description

Start the interrupts running. After this call the specified hardware interrupts (IRQ) is enabled. Any characters received on any of the ports installed with SerialInstallPort will be placed into the receive buffers for

that port. Any characters sent to an installed port will be transmitted. Interrupts may be stopped at any time by calling SerialStop. This function should be called to initiate data transfer.

Example

```
result = SerialStart()
```

Start ***BCTSIO*** interrupts running.

SerialStop

Definition

C int SerialStop(void);
Visual Basic SerialStop() as Integer

Parameters

Return value 0 = Successfully started interrupts
-9 = Interrupts not started

Description

Stop the Serial DLL interrupts running. After this call all the Serial DLL hardware interrupts (IRQ) are disabled Any characters received on any of the ports will be ignored and no more transmission will take place. Interrupts may be started at any time by calling SerialStart. This function should be called before the user program terminates otherwise characters will continue to be received into the internal buffers, unless of course, this is specifically required.

Example

```
result = SerialStop(void);
```

Stop ***BCTSIO*** interrupts running.

SerialTxReady

Definition

C int SerialTxReady(int PortNo);
Visual Basic SerialTxReady(ByVal PortNo As Integer) As Integer

Parameters

PortNo The port number returned from SerialInstallPort when installing the port
Return value 0 The transmitter is not ready for another character
1 The transmitter is ready for another character
-2 PortNo is invalid

Description

Check to see if the transmitter for the specified port is ready to transmit another character. If the transmitter is not busy then another character may be sent immediately.

Example

```
result = SerialTxReady(7)
```

Check to see if the transmitter on port 7 is ready to transmit another character.

SerialTxEmpty

Definition

C int SerialTxEmpty(int PortNo);
Visual Basic SerialTxEmpty(ByVal PortNo As Integer) As Integer

Parameters

PortNo The port number returned from SerialInstallPort when installing the port
Return value 1 The transmitter is completely empty of any characters
0 The transmitter still has characters to transmit
-2 PortNo is invalid

Description

Check to see if the transmitter for the specified port is completely empty of characters, i.e. all characters have been sent. This function is designed

to be used in RS485 (half duplex) systems where the transmitter is to be turned off after the last character has been sent.

Example

```
result = SerialTxEmpty(7)
```

Check to see if the transmitter on port 7 is empty of all characters.

SerialVersion

Definition

C int SerialVersion(void);
Visual Basic SerialVersion() as Integer

Parameters

Return value Return the ***BCTSIO*** version which is 0x100 or &H100 (100H) for this version of ***BCTSIO***.

Description

Return the current version of ***BCTSIO*** in the form xxyy (hex) where xx is the major version number and yy is the minor version number. For example 0100 (hex) refers to version 1.00.

Example

result = SerialVersion()

Return the version of ***BCTSIO***.

SerialWriteModemControl

Definition

C int SerialWriteModemControl(int PortNo, int Mdata);
Visual Basic SerialWriteModemControl(ByVal PortNo As Integer, ByVal Mdata As Integer) as Integer

Parameters

PortNo The port number returned from SerialInstallPort when installing the port

Mdata The data to output to the Modem Control lines, formatted as follows:

Bit	
0	Complement of the DTR output
1	Complement of the RTS output
2	Complement of the OUT1 output
3	Complement of the OUT2 output (Interrupt Enable)
4	Loopback enable bit

Return value

0	Success
-2	PortNo is invalid

Description

Program the contents of the Modem Control Register to the specified value.

The DTR and OUT1 outputs are used in serial cards to enable and disable the transmit line driver. It performs no function with the RS232 based cards.

The OUT2 output enables the interrupt output from the port if this bit is set to a 1. If this bit is set to 0 all further interrupts from this port will stop.

The loopback enable bit is usually set to 0. If set to 1 the port internally loops back the data and modem lines.

For further details refer to the data sheets on the serial card you are using and the manufacturers data sheets for the 8250 (or equivalent) UART I.C.

The current settings of the modem control register may be read by calling the SerialReadModemControl function.

Example

```
result = SerialWriteModemControl(2, 8)
```

Write 8 to the Modem Control Register for PortNo 2.

SerialReadModemControl

Definition

C int SerialReadModemControl(int PortNo);

Visual Basic SerialReadModemControl(ByVal PortNo As Integer) as Integer

Parameters

PortNo The port number returned from SerialInstallPort when installing the port

Return value >=0 The contents of the Modem Control Register
-2 PortNo is invalid

Description

Read the contents of the Modem Control Register.
See the description for SerialWriteModemControl.

Example

```
result = SerialReadModemControl(2)
```

Read the contents of the Modem Control Register on port 2.

Return Values and Error Codes

The following values can be returned from the functions:

Value	Description
0	Success
-1	Function Failed
-2	Invalid Port
-3	Too many ports
-4	Port in use
-5	Out of memory
-6	Interrupts running
-7	Port not opened
-8	Already started
-9	Not started
-10	Invalid baud rate
-11	Invalid parity
-12	Invalid stop bits
-13	Invalid bits per character
-14	Invalid interrupt (IRQ) number
-15	Invalid interrupt status register address
-16	Interrupt Status Register Address Not set

SIOTEST

SIOTEST is a menu driven program, written in Visual Basic V3.00, designed to demonstrate the use of ***BCTSIO*** and to aid in the testing of the serial port hardware.

SIOTEST allows the user to exercise all the functions of ***BCTSIO*** and monitor the data transfer on all of the serial ports without the need for any programming.

The complete source code for SIOTEST is supplied which provides a complete set of examples showing the usage of each ***BCTSIO*** function. This may also provide a starting point for your own project.

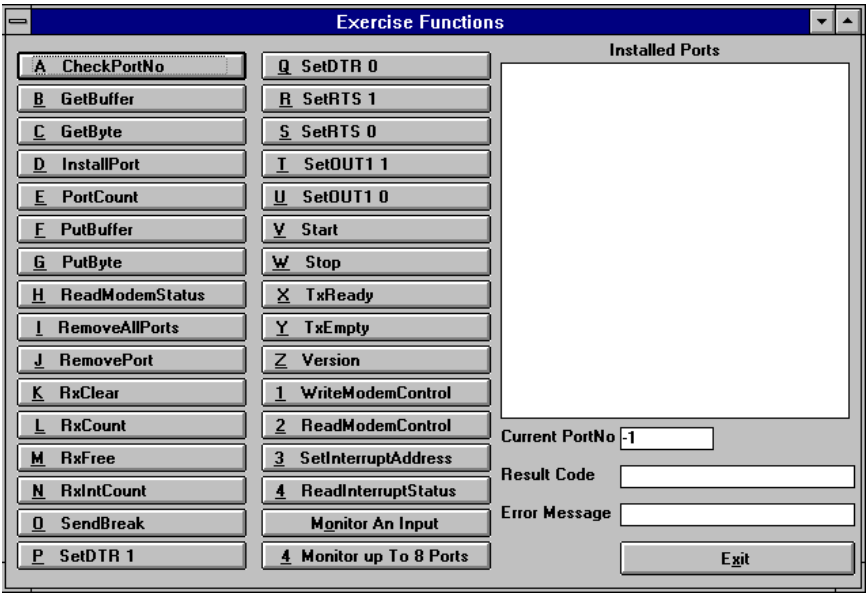
Running SIOTEST

To run SIOTEST either double click on the SIOTEST icon or type use the FILE/RUN/BROWSE menu from the Windows Program Manager to locate SIOTEST.EXE and enter return to execute.

The window on the following page will be displayed:

Select 'Exercise Functions' to test the ***BCTSIO*** functions or 'Exit' to return to Windows.

On selecting the 'Exercise Functions' option the following window will be displayed:



From this screen all of the ***BCTSIO*** functions may be exercised interactively.

The Exercise Functions screen contains the following controls:

Buttons A - Z , 1,2,3 & 4	Exercise each of the ***BCTSIO*** functions
Result Code Text Box	Shows the value returned after calling any function
Error message Text Box	Show the error message represented by the value in the Result Code Text Box
Installed Ports List Box	List of the all of the ports installed with SerialInstallPort
Current PortNo Text Box	The PortNo, selected from the Installed Ports List Box, which will be used as the PortNo parameter for the ***BCTSIO*** functions.
Port Monitor Input Button	Allows the monitoring of any characters received on the currently selected PortNo. Any characters typed are transmitted.
Exit Button	Return to the Main SIOTEST menu

SIOTEST Functions

Before any of the other SIOTEST functions may be used successfully at least 1 port must be installed by using SerialInstallPort and the ***BCTSIO*** interrupt system must be started by using SerialStart. (Note: SIOTEST does not enforce the order in which functions are exercised,, even if this is invalid, so that the operation of all functions may observed if required)

A SerialCheckPortNo

SerialCheckPortNo is called to check to see if the PortNo shown in the Current PortNo text box is valid and has been installed using SerialInstallPort.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

B SerialGetBuffer

The following window is displayed:

The controls on the GetBuffer window as follows:

PortNo Text Box	The currently selected PortNo
Received Buffer	Any bytes read are placed into here
SerialPutBuffer Result Text Box	The result of calling SerialGetBuffer
GetBuffer Button	Call SerialGetBuffer
Exit Button	Return to Test Functions window

when the GetBufferButton is 'pressed' SerialGetBuffer is called any characters returned are copied into the Received Buffer. The result of the call to SerialGetBuffer is placed in the SerialGetBuffer Result text box.

C SerialGetByte

Read the next byte from the currently selected port.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

D SerialInstallPort

A window is displayed:
The controls on the Install Port window are as follows:

PortNo Address	The base address for the port in hexadecimal E.g.: COM1 would be 03F8
Irq	The hardware interrupt number (0-15)
Baud Rate	The Baud rate 110-115000
Stop Bits	The number of stop bits (1 or 2)
Parity	The Parity 0=None, 1=Odd, 2=Even
Buflen	The length of the receive buffer
Result	The result of the call to SerialInstallPort
SerialInstallPort button	Call SerialInstallPort
Exit button	Return to the functions menu

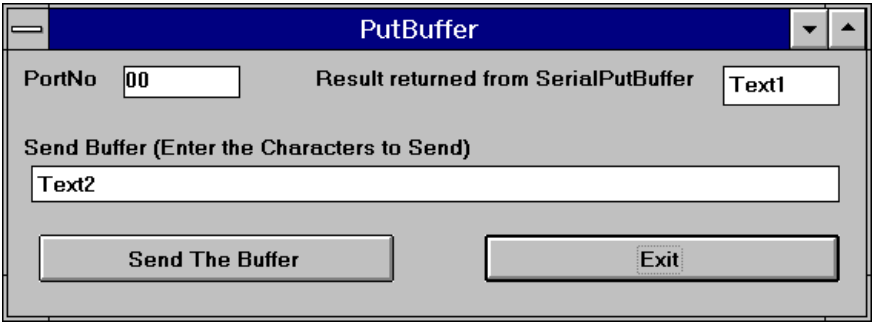
Enter the details into the fields, the default is set for COM1:. Then 'press' the SerialInstallPort button, the result will be displayed in the Result field. If successful the PortNo will be placed into the result field and the port details will be added to the Port List on the functions screen.

E SerialPortCount

SerialPortCount is called to see how many ports are currently installed in ***BCTSIO***.
If this value is not equal to the number of ports shown in the Ports Installed List Box it is possible that another task has installed some ports in ***BCTSIO***.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

F SerialPutBuffer

The following window is displayed:



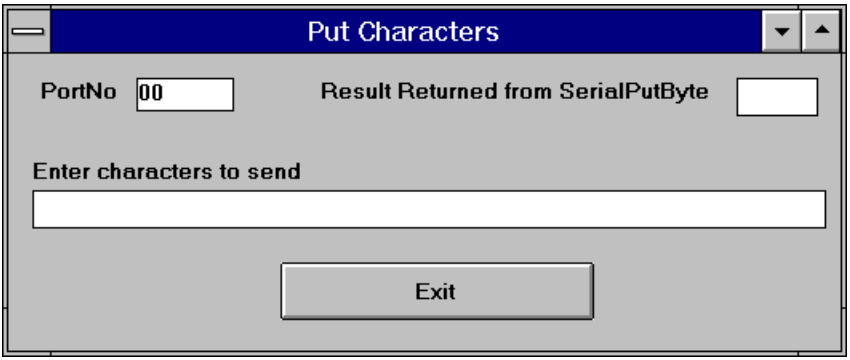
The controls on the PutBuffer window are as follows:

PortNo Text Box	The currently selected PortNo
Characters to Send Buffer	Any characters to be sent are placed in here.
SerialPutBuffer Result Text Box	The result of calling SerialPutBuffer
PutBuffer Button	Call SerialPutBuffer
Exit Button	Return to Test Functions window

When the PutBuffer button is 'pressed' SerialPutBuffer is called to transmit any characters that have been typed into the send buffer. The result of the call to SerialPutBuffer is placed in the SerialPutBuffer Result text box.

G SerialPutByte

The following window is displayed:



The controls on the PutBuffer window are as follows:

PortNo	The currently selected PortNo
Characters to Send	Any characters typed in here will be sent
SerialPutByte Result	The result of calling SerialPutByte
Exit Button	Return to Test Functions window

When a character is placed in the character send buffer it will be immediately transmitted out of the port shown in the PortNo field by calling SerialPutByte. The result of calling SerialPutByte will be placed into the result text box.

H SerialReadModemStatus

Read the state of the modem control lines (inputs) which are reflected in the modem status register.

I SerialRemoveAllPorts

Remove all the previously installed ports from operating under ***BCTSIO***.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

J SerialRemovePort

Remove the previously installed port Portno from operating under ***BCTSIO***. The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

K SerialRxClear

Delete all of the characters from the receive buffer for the specified port.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

L SerialRxCount

Return the number of characters ready to be read from the receive buffer for the specified port.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

M SerialRxFree

SerialRxFree is called to see how many free bytes are available in the receive buffer.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

N SerialRxIntCount

Return the number of receiver interrupts, which is equal to the number of characters received for all ports in total, since the last call to SerialStart.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

O SerialSendBreak

Transmit a break to the specified port. Normally the transmit data line is sending a 'marking' signal indicating no data is present, A break is a 'spacing' signal which lasts for longer than a valid character pattern.

P SerialSetDTR (1)

Send 1 to the DTR line on the port shown in the Port Select Box.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

Q SerialSetDTR (0)

Send 0 to the DTR line on the port shown in the Port Select Box.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

R SerialSetRTS (1)

Send 1 to the RTS line on the port shown in the Port Select Box.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

S SerialSetRTS (0)

Send 0 to the RTS line on the port shown in the Port Select Box.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

T SerialSetOut1(1)

Send 1 to the OUT1 line on the port shown in the Port Select Box.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

U SerialSetOut1(0)

Send 0 to the OUT1 line on the port shown in the Port Select Box.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

V SerialStart

Start the ***BCTSIO*** interrupts running. After this call the specified hardware interrupts (IRQ) is enabled. Any characters received on any of the ports installed with SerialInstallPort will be placed into the receive buffers for that port. Any characters sent to an installed port will be transmitted. Interrupts may be stopped at any time by calling SerialStop. This function should be called to initiate data transfer.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

W SerialStop

Stop the ***BCTSIO*** interrupts running. After this call all the ***BCTSIO*** hardware interrupts (IRQ) are disabled Any characters received on any of the ports will be ignored and no more transmission will take place. Interrupts may be started at any time by calling SerialStart. This function should be called before the user program terminates otherwise characters will continue to be received into the internal buffers, unless of course, this is specifically required.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

X SerialTxReady

Check that the transmitter for the specified port is ready to receive another character.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

Y SerialTxEmpty

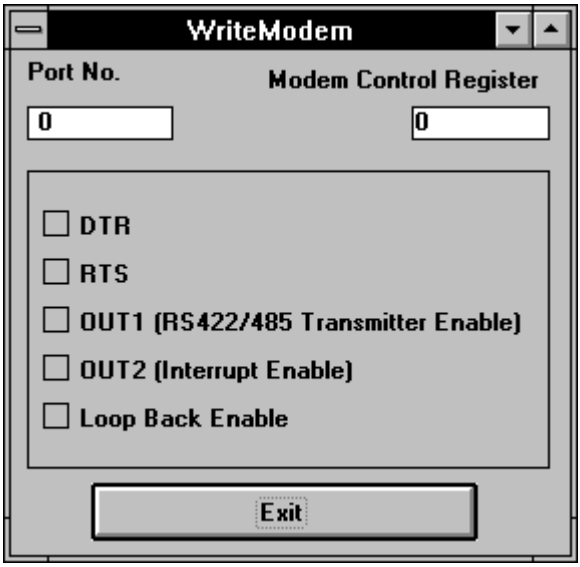
Check to see if the transmitter is completely empty of all characters.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

Z SerialVersion

Return the version of ***BCTSIO***.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

1 SerialWriteModemControl

The following window is displayed:



The controls on the Modem Control Window are as follows:

PortNo	The currently selected PortNo
Modem Control Register	The result of calling SerialReadModemControl
DTR Check Box	Set/Reset the DTR Line
RTS Check Box	Set/Reset the RTS Line
OUT1 Check Box	Set/Reset the OUT1 Line
OUT2 Check Box	Set/Reset the OUT2 Line
Loopback Enable Check Box	Set/Reset the Loopback feature
Exit Button	Return to Test Functions window

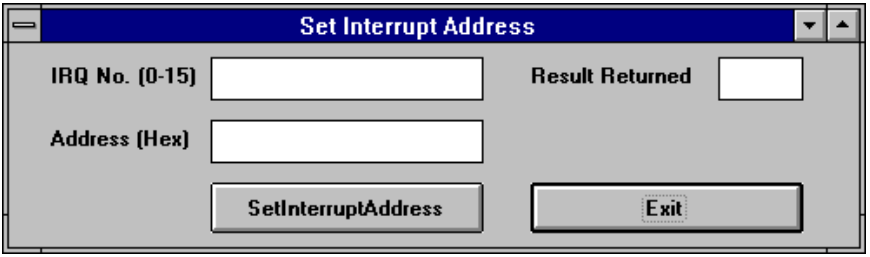
The Modem Control Window allows full interactive control of the Modem Control Register. Each 'click' in the check boxes will invert the state of that line which will be output to the Modem Control Register. The current setting of the Modem Control Register will be read back and displayed in the Modem Control Register field.

2 SerialReadModemControl

Read the contents of the Modem Control Register.
The return value is shown in the Result Code Text Box.
The Error Message, if any, is shown in the Error Message Text Box

3 SetInterruptAddress

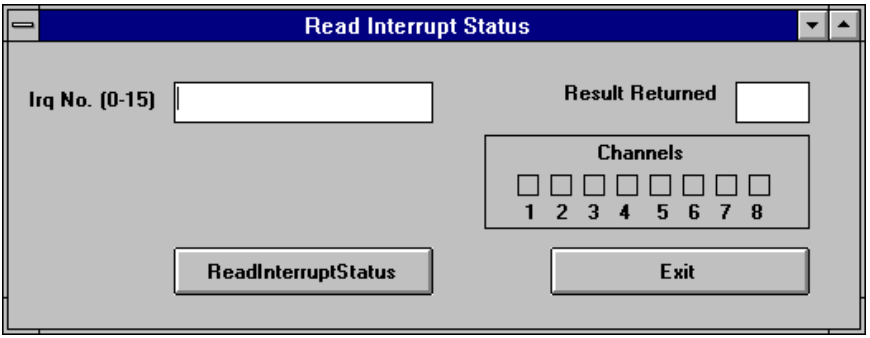
The following window is displayed



This windows allows the address of the Interrupt Status Register to be set and assigned to a particular IRQ number. This is only necessary when more than one port uses the same interrupt line which is only supported on the MSC 4/6/8 range of communications adapters.

4 ReadInterruptStatus

The following window is displayed



If an Interrupt Status Register has been assigned to a particular IRQ then the interrupt status may be displayed by selecting the corresponding IRQ

number. The value returned is shown as a number and if valid (>= 0) the interrupt status for each channel is shown.

SYSTEM REQUIREMENTS

Hardware:

BCTSIO requires that the hardware is compatible with the following Blue Chip Technology communication adapters:

- MSC8 2, 4, 6 and 8 channel versions
- SIO2
- SPIO
- SIO2P
- SIO4d
- Any IBM PC compatible Asynchronous Communications Adapter using the INS8250 UART or equivalent and that pin OUT2 is used to enable the interrupt output.
- IBM-PC/AT -386 and 100% compatibles that are running Microsoft Windows 3.1.

To share an interrupt between multiple ports the hardware must have been specifically design to do so, e.g. the Blue Chip Technology MSC8 card.

Software:

- ***BCTSIO*** is designed to operate under Windows version 3.1 in enhanced mode.

Performance Considerations

The performance of any Windows program, including ***BCTSIO*** for Windows is heavily dependent upon the following factors:

PC Make and Model	Some systems are better than others 32 bit buses are faster than 16 bit buses
CPU	Pentiums are faster than 486s etc.
CPU Clock speed	The faster the better
Amount of Memory	The more the better (At least 4Mb)
Size and Speed of Hard Disk Drive	The faster and bigger the better

Size and type of Swap File	Permanent Swap files are better than temporary ones. Use the size that Windows recommends
Other running Programs	The more programs running the slower the performance of any individual program.

The following items will also affect the performance of the Serial DLL for Windows:

Number of Serial ports supported	The less ports, the higher the performance
Serial baud rates	The faster the baud rates the lower the performance
Sharing Hardware Interrupts	If possible assign each serial port to an individual hardware interrupt (IRQ).
Serial Port Uart Type	Some of the newer UARTs types have build in FIFOs which will increase performance. Check your hardware manual
Receive Buffer Size	In general increasing the buffer size will improve the performance
User program - Receive buffers	Remove the characters from the receive buffers as quickly as possible.
User program - Disk Access	Disk access is slow. Ensure that there is enough room in the receive buffers to receive any incoming characters when a disk access is in progress. If possible stop serial data transfer during this time using some form of flow control.

User program - Screen Access	In general screen access is slow, separate the receive data routines from any screen access routines.

SUMMARY OF THE DISTRIBUTION DISK

The ***BCTSIO*** package is supplied on one 3½" 1.44 Mbyte HD IBM-PC/AT compatible double sided high density diskette.

The following files are supplied:

BCTSIO.DLL	The Serial DLL for Windows library. Placed in the windows\system directory

Source code for SIOTEST (Visual Basic)

BCTSIO.TXT	Serial DLL declarations
GETBUFFE.FRM	Get Buffer Form
INSTPORT.FRM	Install Port form
MAINMENU.FRM	Main Menu
MONINPS .FRM	Monitor up to 8 Inputs form
MONINPUT.FRM	Monitor a single input
PUTBUFFE.FRM	Put a buffer form
PUTCHARS.FRM	Put a character form
TESTFNS.FRM	Test each functions menu
WRITEMOD.FRM	Write modem control register form
SIOTEST.EXE	Executable version of SIOTEST
BCTSIO.H	C Header file for the Serial DLL
VBRUN300.DLL	Visual Basic Runtime library
READ.ME	Last minute documentation update (if present)

DISK FILE	DESCRIPTION	UNCOMPRESSED FILE NAME
SIOTEST.EX_	Executable version of SIOTEST	SIOTEST.EXE
BCTSIO.H_	C Header file for the Serial DLL	BCTSIO.H
BCTSIO.TX_	Serial DLL declarations for Visual Basic	BCTSIO.TXT
GETBUFFE.FR_	Get Buffer Form	GETBUFFER.FRM
INSTPORT.FR_	Install Port form	INSTPORT.FRM
MAINMENU.FR_	Main Menu	MAINMENU FRM
MONINPS.FR_	Monitor up to 8 Inputs form	MONINPS.FRM
MONINPUT.FR_	Monitor a single input	MONINPUT.FRM
PUTBUFFE.FR_	Put a buffer form	PUTBUFFE.FRM
PUTCHARS.FR_	Put a character form	PUTCHARS.FRM
READINT.FR_	Read the interrupt status register	READINT.FRM
SETINT.FR_	Set the address of the interrupt status register	SETINT.FRM
SIOTEST.MA_	Visual Basic project file for SIOTEST	SIOTEST.MAK
TESTFNS.FR_	Test each functions menu	TESTFNS.FRM
WRITEMOD.FR_	Write modem control register form	WRITEMOD.FRM
BCTSIO.DL_	The Serial DLL	BCTSIO.DLL
SETUP.EXE	Set-up program	
VER. DL_	DLL to go with the set-up program	VER .DLL
SETUPKIT.DL_	DLL to go with the set-up program	SETUPKIT.DLL
VBRUN300.DL_	Visual Basic DLL to go with Set-up and SIOTEST	VBRUN300.DLL
DDEML.DL_	DLL to go with the set-up program	DDEML.DLL
SETUP.LST	List of files that makeup the set-up utility	
SETUP1.EX_	Executable part of the set-up utility	

NOTES ABOUT SPECIFIC CARDS

FEATURE	SIO-2	MCS-4/6/8	SPIO (SIO2-P)	SIO-4d
No. of serial channels	2	4/6/8	2	4
RS232	Yes	Yes *	Yes	Yes
RS422	Yes	Yes *	Yes	Yes
RS485	Yes	Yes *	Yes	Yes
20 mA Loop	Yes	Yes *	Yes	Yes
RS485 Transmitter Enable	DTR or OUT1	DTR or OUT1	DTR or OUT1	DTR or OUT1
No. of ports sharing a single interrupt	1	Up to 8 †	1	1

* Using plug in personality modules.
† Using the 'Common Interrupt' Mode